Written by: Aliya Ware
Team: //TODO

Programmer Documentation

This file contains snippets of code from each major component of our project

'''
code: This is the algorithm driving Traveler. Prim's Algorithm is implemented
utilizing a min-heap structure on an adjacency list.

group: //TODO
author(s): Thomas Joyce
last modified: 26 Oct 2021
'''

```
from collections import defaultdict

class Edge:
  def __init__(self, val=None, a_vertex=None, b_vertex=None):
    self.weight = val
    self.a = a_vertex
    self.b = b_vertex

class EdgeMinHeap:
  def __init__(self):
    self.heap = []


  def heapify(self, index):
    minimum = index
    left = 2 * index + 1 # left(node) index
    right = 2 * index + 2 # right(node) index
    # value at left is minimum ?
```

```
    if left < len(self.heap) and self.heap[left].weight < self.heap[index].weight:
        minimum = left
    if right < len(self.heap) and self.heap[right].weight < self.heap[minimum].weight:
        minimum = right
    if minimum != index:
        self.interchange_vertex(index, minimum)

def insert(self, edge):

    if len(self.heap) == 0:
        self.heap.append(edge)
    else:
        self.heap.append(edge)

        for i in range((len(self.heap)//2)-1, -1, -1):
            self.heapify(i)



def delete(self):
    self.interchange_vertex(0, len(self.heap)-1) # Exchange 0th index with last index
    min_edge = self.heap.pop() # pop last element
    for i in range((len(self.heap)//2)-1, -1, -1):
        self.heapify(i)
    return min_edge

def interchange_vertex(self, index_a, index_b):
    temp_val = self.heap[index_a].weight
    temp_a = self.heap[index_a].a
    temp_b = self.heap[index_a].b
    self.heap[index_a].weight = self.heap[index_b].weight
    self.heap[index_a].a = self.heap[index_b].a
    self.heap[index_a].b = self.heap[index_b].b
    self.heap[index_b].weight = temp_val
```

```
        self.heap[index_b].a = temp_a
        self.heap[index_b].b = temp_b

class Graph:
    def __init__(self, v_count):
        self.V = v_count
        self.graph = defaultdict(list)
        self.min_heap = EdgeMinHeap()

def solve(matrix):
    '''
    function: builds a graph using input matrix
    input: list[list]
    ex: [[0, 454639, 716226], [455412, 0, 795474], [717739, 811274, 0]]
    output: list[list] #this is the MST path order.
    ex: [[0,1],[1,2],[2,3],[3,4]]
    '''

    g = Graph(len(matrix))
    #print(g.V)

    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
            if j != i:
                #print("adding: ", i, j, matrix[i][j])
                g.add_edge(i, j, matrix[i][j])
    return g.mst_order()

"""
if __name__ == "__main__":
    g = Graph(4)
    g.add_edge(0, 1, 6)
    g.add_edge(0, 2, 1)
```

```
  g.add_edge(0, 3, 2)
  g.add_edge(1, 2, 5)
  g.add_edge(1, 3, 3)
  g.add_edge(2, 3, 4)
  print("MST order:", g.mst_order())
"""
```

```
""" (Snippet)
Filename: app.py

Purpose:
The main application file for the Flask server.
Contains an endpoint '/get_order' that takes a list of locations, parses them through the google distance matrix api,
creates an adjacency matrix from those distances, parses the matrix through the algorithm, and returns the result.

Authors: Jordan Smith
Group: //Todo
Last modified: 10/29/21

"""

import flask
import json
import urllib
import requests
from login import login_page
```

```
from key import API_KEY
import Prims

app = flask.Flask(__name__)
app.register_blueprint(login_page)

###
#   Globals
###
base_url = "https://maps.googleapis.com/maps/api/distancematrix/json?"

"""
to_url_string
  Converts the given list of locations to a string with url-encoding
  url-encoding translates " " -> "%20", "," -> "%2C", etc.
  We seperate the addresses with "|"
"""

def to_url_string(addrs):
  return urllib.parse.quote_plus("|".join(addrs))
"""
to_adj_mat
  Converts a given dictionary (from Google Distance Matrix API)
  into an adjacency matrix and returns that matrix
"""

def to_adj_mat(data):
  result = []
  for i, row in enumerate(data['rows']):
     result.append([])
     for elem in row['elements']:
        # Each element has distance and time keys
        # Both of those have a value (the raw value in km/sec),
```

```
        # and a formatted value (123 Km or 1 hr 20 min)
        result[i].append(elem["distance"]["value"])

  return result
```

```
"""
Filename: HomeComponent.js


Purpose:
The main application file to construct the home page and insert google maps' autocomplete
feature into the website.
Contains the google maps api key and the google rendering component


Authors: Tammas Hicks
Group: //Todo
Last modified: 10/10/21
"""

  import React, { Component } from "react";
  import { render } from "react-dom";
  import { GoogleComponent } from
  "react-google-location";


  class HomeComponent extends Component {
   constructor(props) {
     super(props);
     this.state = {
       place: null,
     };
   }
```

```
  render() {
    return (
      <div>
        <GoogleComponent
          apiKey={GOOGLEMAPS_API_KEY}
          language={"en"}
          country={"country:in|country:us"}
          coordinates={true}
          locationBoxStyle={"custom-style"}
          locationListStyle={"custom-style"}
          onChange={(e) => {
            this.setState({ place: e });
          }}
        />
      </div>
    );
  }
}


  export default HomeComponent;
```

""" (Snippet)
Filename: MyDirectionsRenderer.js

This component is used to render directions on a google maps component
Written by Tammas Hicks
Team //TODO
last modified on 10/29/21

“””

```
import React from "react";
// this is so we can use the react namespace
```

```
import {
  GoogleMap,
  useLoadScript,
  DirectionsService,
  DirectionsRenderer,
} from "@react-google-maps/api";
import styled from "styled-components";
// imported styled to do css work in the same file. Eventual refactoring should include this in all files.

const libraries = ["places", "directions"];
// this is how we get the map to map to render places and directions

const mapContainerStyle = {
  // we want the map to fill its entire container. We can't use a styled div, because it's an imported component
  width: "100%",
  height: "100%",
  borderRadius: "30px",
};
const center = {
  // centered over eugene. Eventually this should be changed into a prop that accepts user queried location via props
  lat: 44.052071,
  lng: -123.086754,
};
```

"""
Filename: CreateAccount.js

Purpose:
This file formats the user login/signup page. It is able to take in user input and scans the database for existing accounts. It can take users from sign in page to the login page through a “create account button”
It will set up users and send that information to the backend for processing and authentication.

It is secured by a hash key

Authors: Evan Paraiser
Group: //Todo
Last modified: 10/24/21
"""

```
import { useState } from "react";
import { Link } from "react-router-dom";
import classes from "../CSS/CreateAccount.module.css";
import { useHistory } from "react-router";

function CreateAccount() {
  const [userName, setUsername] = useState(null);
  const [password, setPassword] = useState(null);
  const [email, setEmail] = useState(null);

  const history = useHistory();

  function usernameChanger(event) {
    setUsername(event.target.value);
  }

  function passwordChanger(event) {
    setPassword(event.target.value);
  }

  function emailChanger(event) {
    setEmail(event.target.value);
  }

  function PostAccount(event) {
    event.preventDefault();
    const accountInfo = {
      method: "POST",
      headers: {
        "Content-Type": "application/JSON",
        Contents: "accountInfo",
```

```
    },
    body: JSON.stringify({
      username: userName,
      email: email,
      password: password,
    }),
  };

  fetch("/create_account", accountInfo).then((response) => {
    if (response.status === 201) {
      history.push("/MainPage");
    } else if (response.status === 409) {
      alert("That user already exists!");
    } else {
      alert("Failed to create profile!");
    }
  });
}
return (
  <div className={classes.body}>
    <div className={classes.container}>
      <form className={classes.form} id="createAccount">
        <h1 className={classes.formTitle}>Create Account</h1>

        <div className={classes.form__inputGroup}>
          <input
            onChange={usernameChanger}
            type="username"
            className={classes.form__input}
            autoFocus
            placeholder="Username"
          ></input>
        </div>
        <div className={classes.form__inputGroup}>
          <input
            onChange={emailChanger}
            type="email"
            className={classes.form__input}
            placeholder="Email Address"
          />
```

```
          </div>
          <div className={classes.form__inputGroup}>
            <input
              onChange={passwordChanger}
              type="password"
              className={classes.form__input}
              placeholder="Password"
            />
          </div>
          <button className={classes.form__button} onClick={PostAccount}>
            Continue
          </button>
          <br />
          <br />
          <br />
          <p className={classes.form__text}>
            <Link className={classes.form__link} to="/SignIn">
              Already have an account? Sign in
            </Link>
          </p>
        </form>
      </div>
    </div>
  );
}

export default CreateAccount;
```